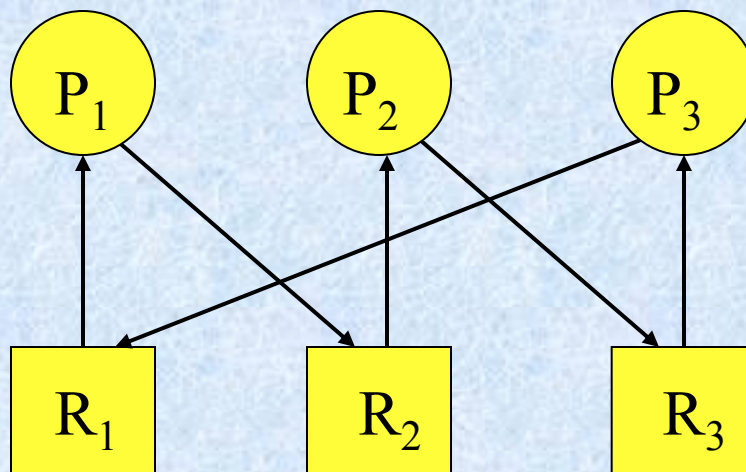


Il problema dello stallo

Stallo (blocco critico; deadlock)

Sospensione irreversibile di **processi**
in competizione per un insieme di **risorse**



Stallo

Si considerano **risorse riusabili**

esempio: *classi di sezioni critiche*

- **Richiesta**

esempio: *wait(mutex)*

ipotesi: richiesta bloccante

- **Assegnazione e utilizzo**

- **rilascio**

esempio: *signal(mutex)*

Stallo

Condizioni per il verificarsi dello stallo

- *Mutua esclusione*
 - risorse non utilizzabili contemporaneamente da più processi
- *Possesso e attesa*
 - i processi che si sospendono mantengono il possesso delle risorse già ottenute
- *Risorse non prerilasciabili*
 - non ammessa la revoca da parte del gestore
 - solo esplicito rilascio da parte dei processi
- *Attesa circolare*

Stallo

Attesa circolare

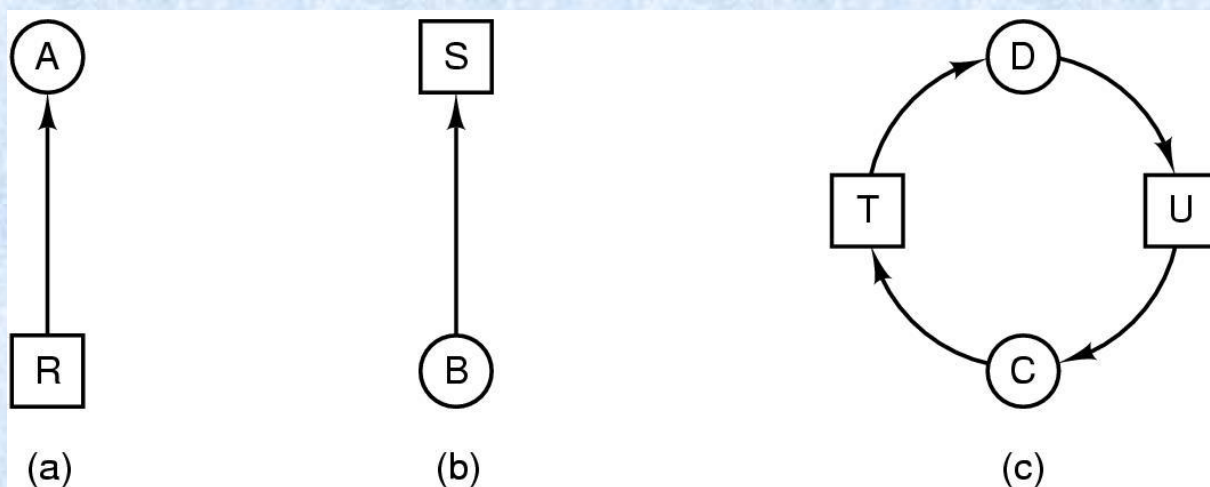
Esiste un insieme \mathcal{P} di processi e un insieme \mathcal{R} di risorse tali che:

- ogni risorsa di \mathcal{R} è assegnata a un processo di \mathcal{P}
- ogni processo di \mathcal{P} è in attesa di una risorsa di \mathcal{R}

--> il verificarsi dell'attesa circolare dipende dall'ordine con cui i processi eseguono richieste e rilasci

Modelli per lo Stallo

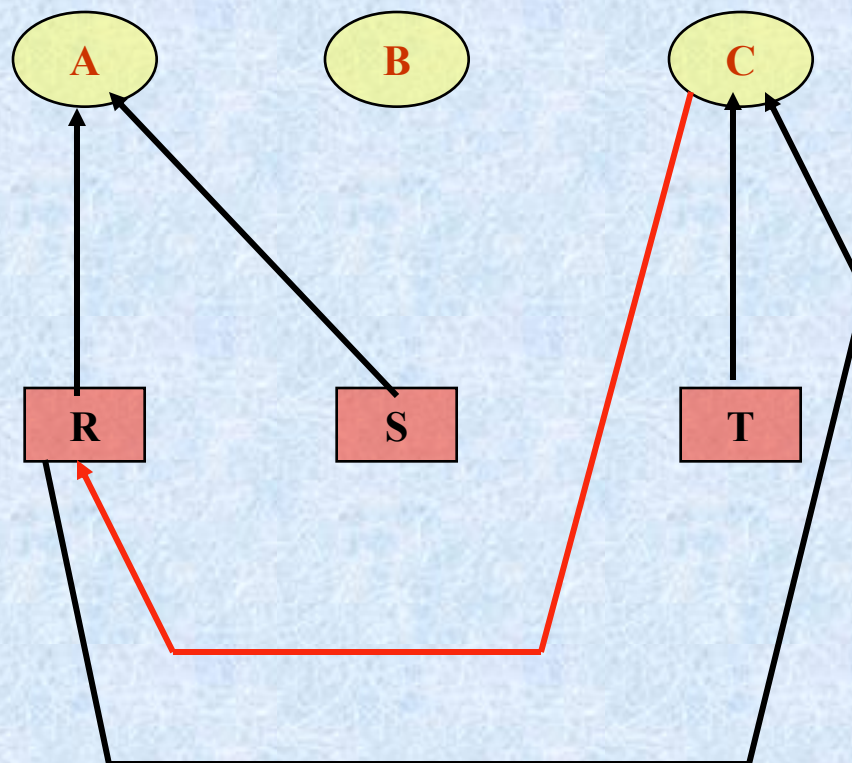
- Grafi di allocazione delle risorse



- La risorsa **R** è assegnata al processo **A**
- Il processo **B** richiede/aspetta la risorsa **S**
- I processi **C** e **D** sono attesa circolare

Sequenza di richieste e rilasci che non provoca stallo

A richiede R
C richiede T
A richiede S
C richiede R
A rilascia R
A rilascia S



Ipotesi: A necessita solo di R e S

Sequenza di richieste e rilasci che provoca stallo

A richiede R

B richiede S

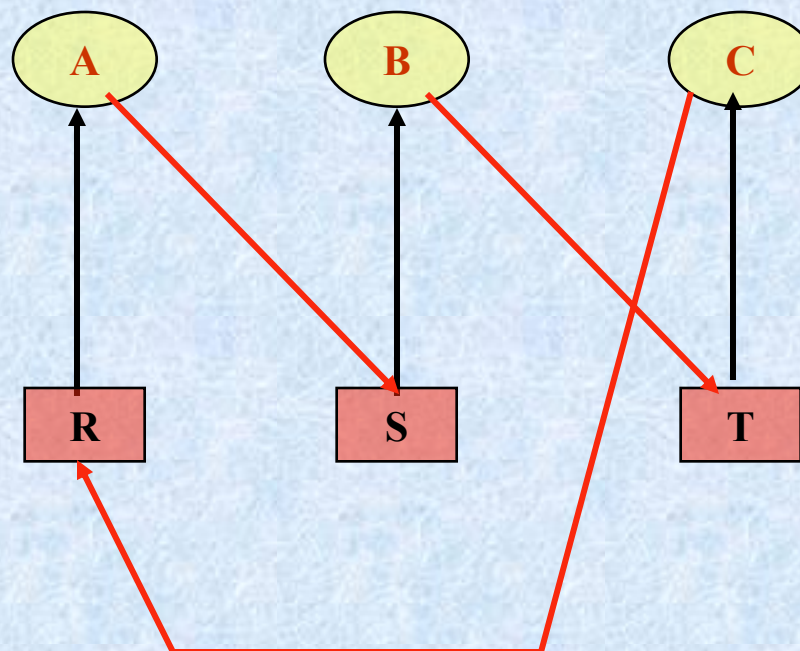
C richiede T

A richiede S

B richiede T

C richiede R

Stallo!



Risorse singole e risorse multiple

Risorse multiple

- Risorse suddivise in tipi
- Per ogni tipo: molteplicità M
numero totale di risorse del tipo
- Disponibilità D : numero di risorse del tipo che sono attualmente disponibili
- Modalità di richiesta:
 - 1) Richiesta singola
 - 2) Richiesta multipla
 - richieste k risorse del tipo
 - se $k \leq D$ assegnate k risorse
 - se $k > D$ nessuna risorsa assegnata

Metodi per il trattamento dello stallo

- Prevenzione statica
- Prevenzione dinamica
- Riconoscimento ed eliminazione
- In realtà molti sistemi (Unix, Windows) non affrontano il problema
 - Ragionevole se:
 - le risorse sono abbondanti e lo stallo si verifica raramente
 - Il costo per evitare lo stallo è troppo elevato

Stallo

Prevenzione statica

La prevenzione statica agisce sulle condizioni che portano allo stallo:

- Mutua esclusione
 - invalidare utilizzando lo spool
- Possesso e attesa
 - invalidare imponendo vincoli sulle modalità di richiesta
unica richiesta multipla per tutte le risorse necessarie
- Attesa circolare
 - invalidare introducendo ordinamento delle risorse

Stallo: prevenzione statica

Metodi di prevenzione

Condizione invalidata	Metodi
Mutua esclusione	Usare spool
Possesso e attesa	Richiedere tutte le risorse inizialmente
Attesa circolare	Ordinare le risorse

Prevenzione statica sulla condizione di mutua esclusione

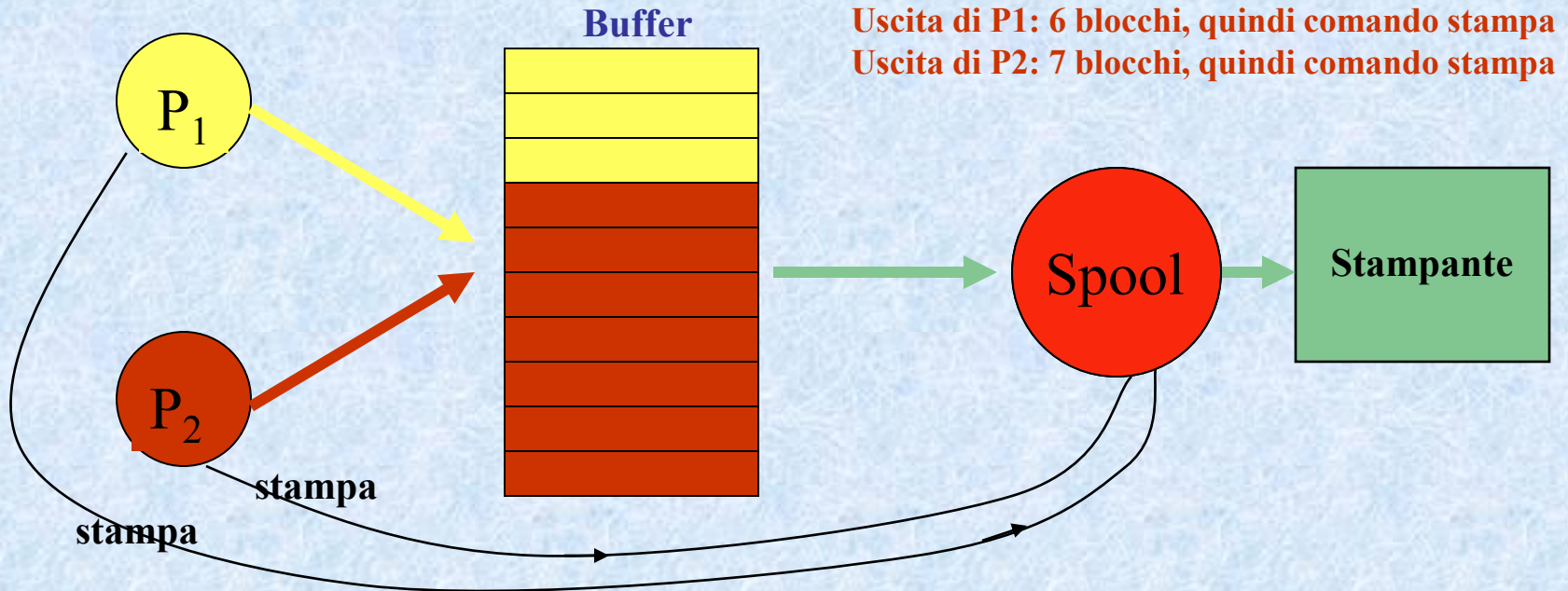
- Alcuni dispositivi (ad esempio le stampanti) possono essere gestiti con *spool*
 - I processi scrivono l'output in un *buffer di spool* (risorsa utilizzata senza il vincolo della mutua esclusione)
 - La stampante fisica non è condivisa (utilizzata solo dal gestore dello spool): quindi lo stallo per la stampante è eliminabile

Ma:

- Non tutti i dispositivi possono essere gestiti con spool
- Ci può essere stallo nell'accesso al buffer di spool
buffer: risorsa multipla

Stallo con risorse multiple: SPOOL

1) Sequenza che non provoca stallo



Al tempo t: P_1 ha inviato 4 blocchi di stampa; P_2 ha inviato 4 blocchi di stampa;

Dopo tempo t:

P_1 invia 1 blocco; P_1 invia 1 blocco e invia comando Stampa

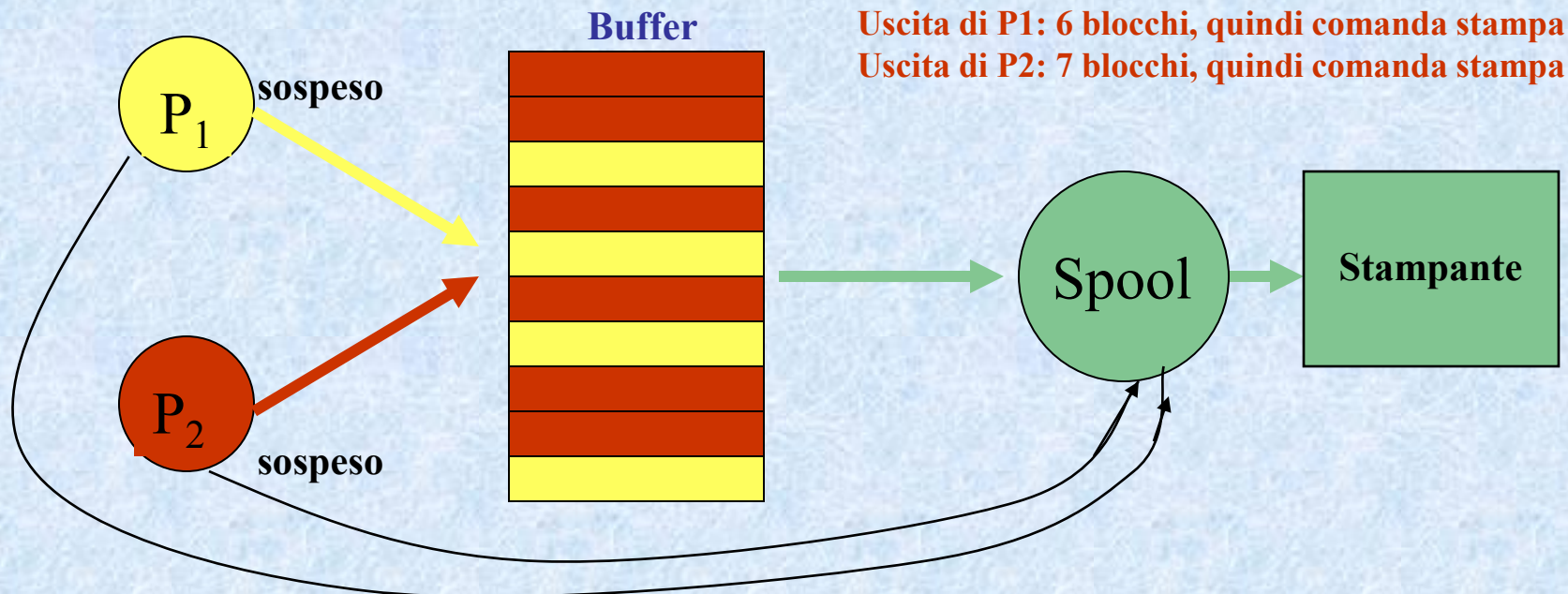
Spool inizia stampa dell'uscita di P_1

P_2 invia 1 blocco; P_2 invia 1 blocco; P_2 invia 1 blocco e invia comando Stampa

Spool completa stampe

Stallo con risorse multiple: SPOOL

2) Sequenza che provoca stallo



Al tempo t: P1 ha inviato 4 blocchi di stampa; P2 ha inviato 4 blocchi di stampa;

Dopo tempo t: P2 invia 1 blocco

P2 invia 1 blocco

P2 invia 1 blocco: P2 sospeso

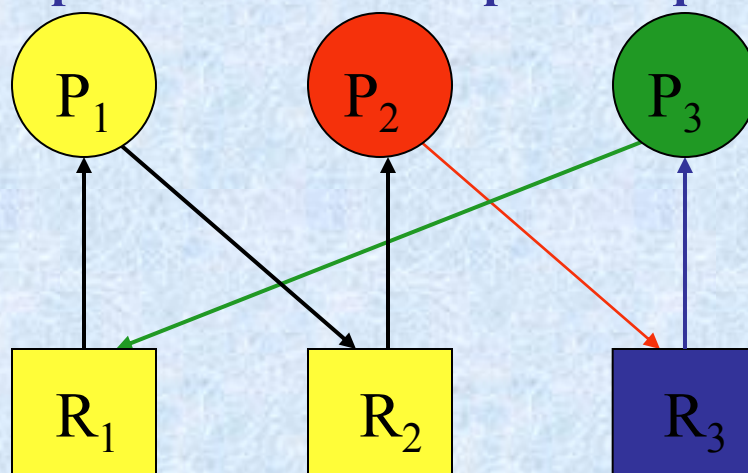
P1 invia 1 blocco: P1 sospeso

Prevenzione statica basata sull'invalidazione della condizione di possesso e attesa

- il processo conosce in anticipo tutte le risorse che utilizzerà
 - le richiede prima dell'utilizzo con una richiesta multipla
 - > se le ottiene, si ha possesso ma non attesa
 - > se non le ottiene, si ha attesa ma non possesso
- Problemi
 - Può non sapere di quali risorse avrà bisogno
 - Vincola risorse che altri processi potrebbero usare
- Variante, senza il vincolo della richiesta iniziale di tutte le risorse
 - Ad ogni nuova richiesta, il processo rilascia tutte le risorse che possiede ...
 - ... e quindi le richiede di nuovo insieme alla nuova richiesta

Prevenzione statica basata sull'invalidazione della condizione di attesa circolare

- Ordinamento delle risorse
 - Le richieste dei processi devono rispettare questo ordinamento



Risorsa di indice massimo

P_3 ha violato il vincolo di richiesta ordinata

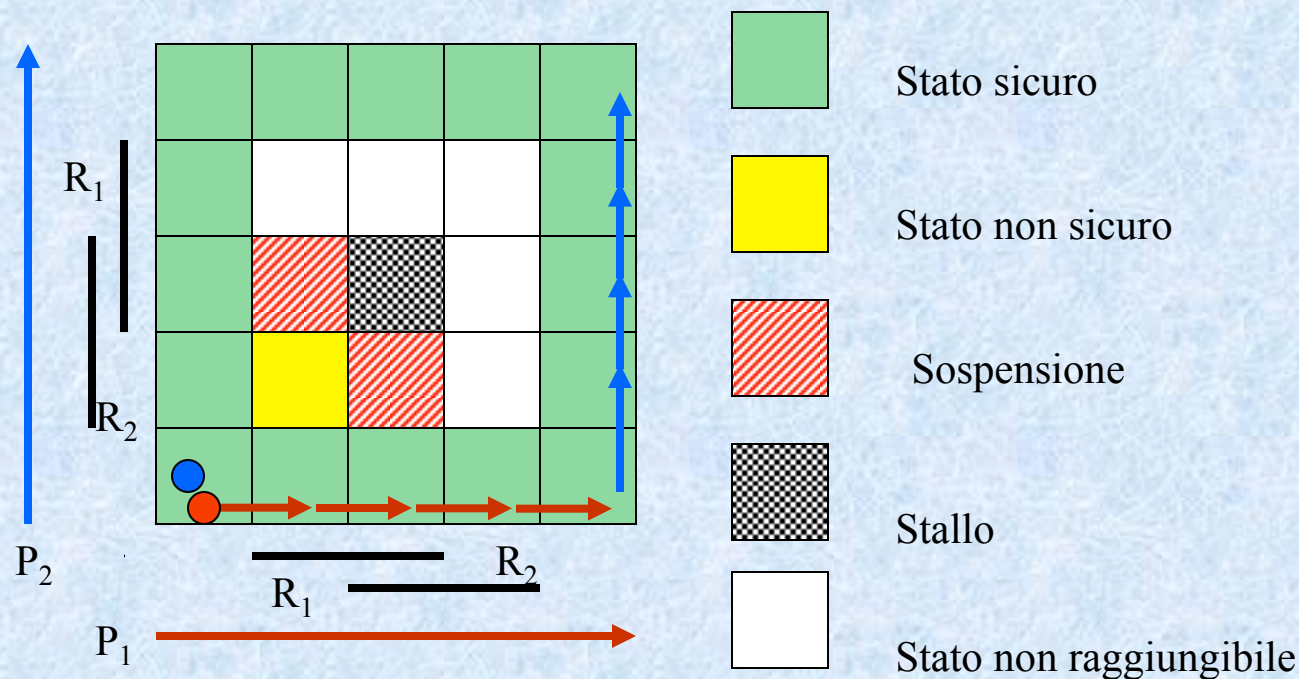
P_2 causa attesa circolare

Prevenzione dinamica

- *banchiere*: gestore delle risorse
- *politica*: “*algoritmo del banchiere*”
 - processo P richiede risorsa R
 - banchiere assegna risorsa solo se l’assegnazione conduce in uno *stato sicuro*
- Lo stato S è sicuro se a partire da S esiste una sequenza di assegnazioni e rilasci che permetta di soddisfare tutte le richieste
==> e quindi di far *terminare* tutti i processi

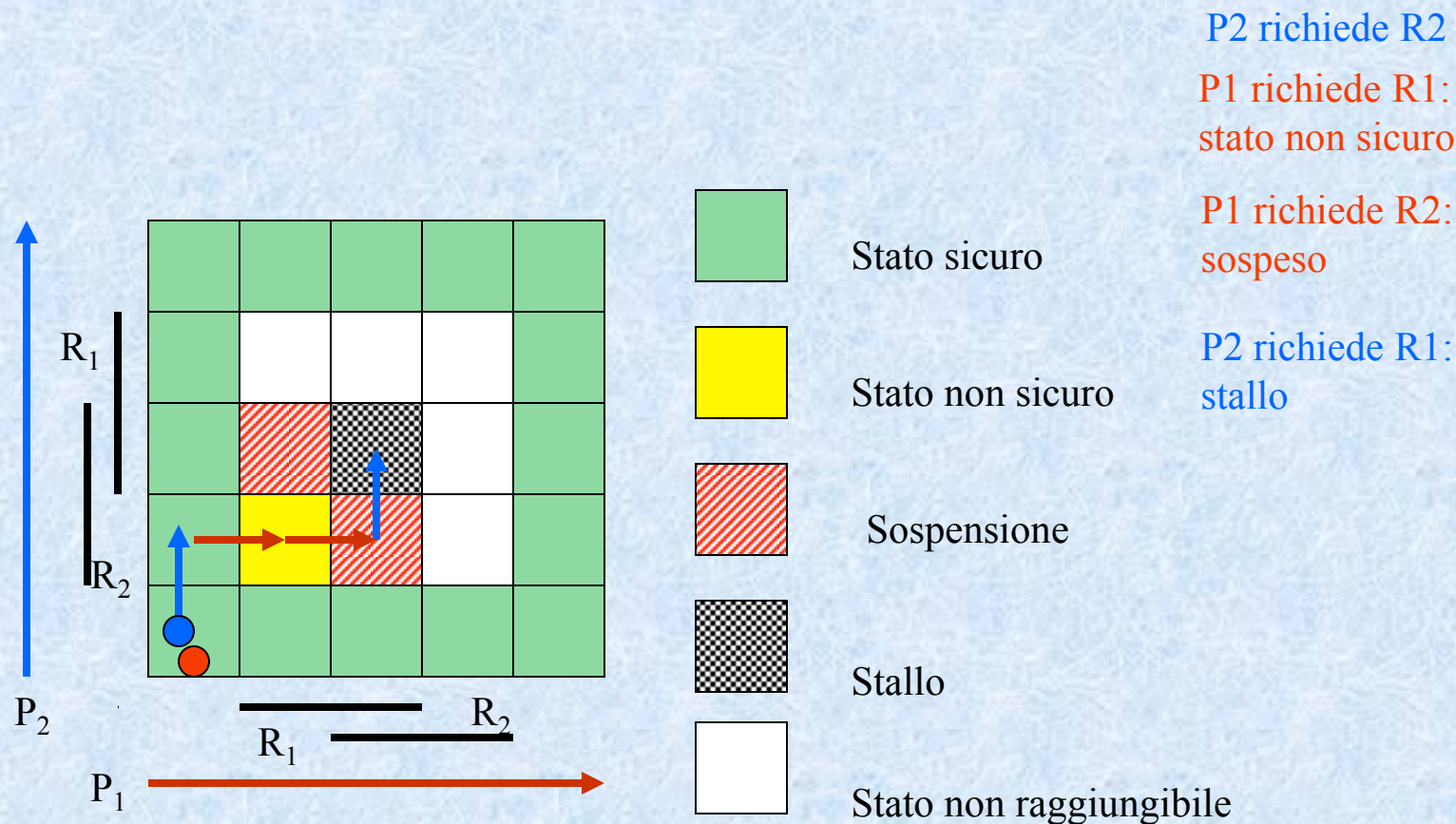
Concetto di Stato Sicuro

1) Evoluzione del sistema per stati sicuri



Concetto di Stato Sicuro

2) Il sistema raggiunge uno stato non sicuro e *può andare* in stallo:



Algoritmo del banchiere

3) Il gestore non soddisfa le richieste che condurrebbero in uno stato non sicuro : P2 richiede R2

P2 richiede R2

P1 richiede R1:
non assegnata;
P1 sospeso

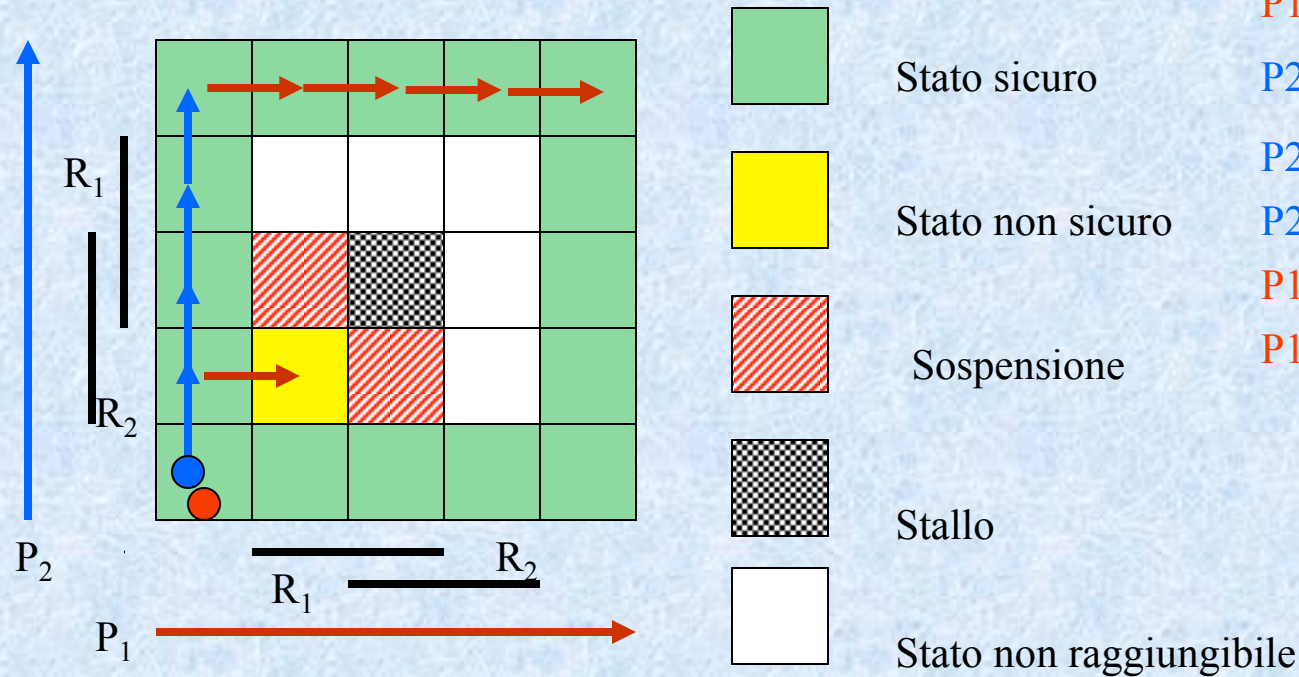
P2 richiede R1:

P2 rilascia R2:

P2 rilascia R1

P1 ottiene R1:

P1 avanza e termina



Algoritmo del banchiere per risorse multiple di un unico tipo

- Prima di iniziare l'esecuzione ogni processo dichiara il massimo numero di risorse che gli sono necessarie
- Ad ogni richiesta di assegnazione l'algoritmo verifica se l'assegnazione della risorsa conduce in uno stato sicuro
 - Si ipotizza l'assegnazione e si considera lo stato **S** così raggiunto
 - per ogni processo si calcolano le unità di risorsa che deve ancora richiedere (**esigenza residua R**)
 - si considerano i processi in ordine di **R** crescente e per ognuno:
 - si verifica se la disponibilità attuale consente di soddisfare **R**;
 - in caso affermativo si ipotizza l'assegnazione, la conseguente terminazione e il rilascio delle risorse assegnate, e si considera lo stato così raggiunto
 - Si ripete il procedimento per tutti i processi non terminati.
 - Se tutti i processi possono terminare lo stato **S** è sicuro
- Se lo stato **S** è sicuro la risorsa viene assegnata; altrimenti il processo viene sospeso in attesa che la disponibilità aumenti fino a quando l'assegnazione porti in uno stato sicuro

Verifica dello stato sicuro per risorse multiple di più tipi

Per ogni risorsa R_k : D : vettore di disponibilità

(D_k numero di unità disponibili della risorsa R_k)

Per ogni processo P_j : A_j : vettore di assegnazione; E_j : vettore di esigenza residua;

$E_j \leq D$ se $E_{jk} \leq D_k$ per ogni k

Inizialmente ogni processo P_j è non marcato

```
while ( $\exists$  processi non marcati) {  
    if ( $\exists P_j$  soddisfa  $E_j \leq D$ ) {  
        Marca il processo  $P_j$ ;  
         $D_j = D_j + A_j$  ;  
    } else termina e segnala lo stallo di tutti i  
    processi non marcati;  
}
```

termina con successo: lo stato è sicuro

Copyright © 2004 - The McGraw-Hill Companies srl

Stato raggiunto dal sistema al tempo t :

Verifica dello stato sicuro:

P. Ancilotti, M. Boari, A. Ciampolini, G. Lipari - Sistemi Operativi

VERIFICA: LO STATO RAGGIUNTO AL TEMPO t E' SICURO (Ver 2)

Verifica dello stato sicuro:

1) l'esigenza di P2 può essere soddisfatta e P2 può terminare

		MOLTEPLICITA' →									
ASSEGNAZIONE ATTUALE →			R1	R2	R3	R4		R1	R2	R3	R4
		P1	2	2	1	1		4	5	5	5
		P2	-	-	-	-					
		P3	0	1	0	0					
		P4	0	2	2	2					
DISPONIBILTA' ATTUALE			R1	R2	R3	R4					
			2	1	2	2					
		ESIGENZA ATTUALE									
			R1	R2	R3	R4					
		P1	0	2	0	0					
		P2	-	-	-	-					
		P3	0	0	0	2					
		P4	0	0	3	0					

Verifica dello stato sicuro:

1) l'esigenza di P2 può essere soddisfatta e P2 può terminare

2) l'esigenza di P3 può essere soddisfatta e P3 può terminare

ASSEGNAZIONE ATTUALE →	MOLTEPLICITA' →					R1	R2	R3	R4	
		R1	R2	R3	R4	4	5	5	5	
	P1	2	1	1	1					
	P2	-	-	-	-					
	P3	-	-	-	-					
	P4	0	2	2	2					
DISPONIBILTA' ATTUALE		R1	R2	R3	R4					
		2	2	2	2					
						ESIGENZA ATTUALE				
							R1	R2	R3	R4
						P1	0	2	0	0
						P2	-	-	-	-
						P3	-	-	-	-
						P4	0	0	3	0

VERIFICA: LO STATO RAGGIUNTO AL TEMPO t E' SICURO (Ver 3)

Verifica dello stato sicuro:

- 1) l'esigenza di P2 può essere soddisfatta e P2 può terminare
- 2) l'esigenza di P3 può essere soddisfatta e P3 può terminare

ASSEGNAZIONE ATTUALE →		MOLTEPLICITA' →									
			R1	R2	R3	R4		R1	R2	R3	R4
		P1	2	1	1	1		4	5	5	5
		P2	-	-	-	-					
		P3	-	-	-	-					
		P4	0	2	2	2					
DISPONIBILTA' ATTUALE			R1	R2	R3	R4					
			2	2	2	2					
			R1	R2	R3	R4					
		P1	0	2	0	0					
		P2	-	-	-	-					
		P3	-	-	-	-					
		P4	0	0	3	0					
		ESIGENZA ATTUALE									

Verifica dello stato sicuro:

- 1) l'esigenza di P2 può essere soddisfatta e P2 può terminare
- 2) l'esigenza di P3 può essere soddisfatta e P3 può terminare
- 3) l'esigenza di P1 può essere soddisfatta e P1 può terminare

ASSEGNAZIONE ATTUALE →		MOLTEPLICITA' →									
			R1	R2	R3	R4		R1	R2	R3	R4
		P1	-	-	-	-		4	5	5	5
		P2	-	-	-	-					
		P3	-	-	-	-					
		P4	0	2	2	2					
DISPONIBILTA' ATTUALE			R1	R2	R3	R4					
			4	3	3	3					
			R1	R2	R3	R4					
		P1	-	-	-	-					
		P2	-	-	-	-					
		P3	-	-	-	-					
		P4	0	0	3	0					
		ESIGENZA ATTUALE									

VERIFICA: LO STATO RAGGIUNTO AL TEMPO t E' SICURO (Ver 4)

Verifica dello stato sicuro:

- 1) l'esigenza di P2 può essere soddisfatta e P2 può terminare
- 2) l'esigenza di P3 può essere soddisfatta e P3 può terminare
- 3) l'esigenza di P1 può essere soddisfatta e P1 può terminare

ASSEGNAZIONE ATTUALE →

	R1	R2	R3	R4
P1	-	-	-	-
P2	-	-	-	-
P3	-	-	-	-
P4	0	2	2	2

DISPONIBILTA' ATTUALE

	R1	R2	R3	R4
	4	3	3	3

MOLTEPLICITA' →

	R1	R2	R3	R4
	4	5	5	5

ESIGENZA ATTUALE

	R1	R2	R3	R4
P1	-	-	-	-
P2	-	-	-	-
P3	-	-	-	-
P4	0	0	3	0

Verifica dello stato sicuro:

- 1) l'esigenza di P2 può essere soddisfatta e P2 può terminare
- 2) l'esigenza di P3 può essere soddisfatta e P3 può terminare
- 3) l'esigenza di P1 può essere soddisfatta e P1 può terminare
- 4) l'esigenza di P4 può essere soddisfatta e P4 può terminare

STATO SICURO !

ASSEGNAZIONE ATTUALE →

	R1	R2	R3	R4
P1	-	-	-	-
P2	-	-	-	-
P3	-	-	-	-
P4	-	-	-	-

DISPONIBILTA' ATTUALE

	R1	R2	R3	R4
	4	5	5	5

MOLTEPLICITA' →

	R1	R2	R3	R4
	4	5	5	5

ESIGENZA ATTUALE

	R1	R2	R3	R4
P1	-	-	-	-
P2	-	-	-	-
P3	-	-	-	-
P4	-	-	-	-

ALGORITMO DEL BANCHIERE CON RISORSE MULTIPLE DI PIU' TIPI (1.2)

Stato (**sicuro**) raggiunto dal sistema al tempo t :

ASSEGNAZIONE ATTUALE →	MOLTEPLICITA' →									
		R1	R2	R3	R4		R1	R2	R3	R4
	P1	2	1	1	1		4	5	5	5
	P2	2	0	1	2					
	P3	0	1	0	0		P1	0	2	0
	P4	0	2	2	2		P2	0	1	0
DISPONIBILTA' ATTUALE							P3	0	0	2
		R1	R2	R3	R4		P4	0	0	3
		0	1	1	0		ESIGENZA ATTUALE			

Il processo P1 richiede una risorsa di tipo R2: stato raggiunto dopo l'ipotetica assegnazione:

ASSEGNAZIONE ATTUALE →	MOLTEPLICITA' →									
		R1	R2	R3	R4		R1	R2	R3	R4
	P1	2	2	1	1		4	5	5	5
	P2	2	0	1	2					
	P3	0	1	0	0		P1	0	1	0
	P4	0	2	2	2		P2	0	1	0
DISPONIBILTA' ATTUALE							P3	0	0	2
		R1	R2	R3	R4		P4	0	0	3
		0	0	1	0		ESIGENZA ATTUALE			

ALGORITMO DEL BANCHIERE CON RISORSE MULTIPLE DI PIU' TIPI (1.3)

Il processo P1 richiede una risorsa di tipo R2: stato raggiunto dopo l'ipotetica assegnazione:

ASSEGNAZIONE ATTUALE →	MOLTEPLICITA' →									
		R1	R2	R3	R4		R1	R2	R3	R4
	P1	2	2	1	1		4	5	5	5
	P2	2	0	1	2					
	P3	0	1	0	0					
	P4	0	2	2	2					
DISPONIBILTA' ATTUALE		R1	R2	R3	R4					
		0	0	1	0					
						</				

Verifica dello stato sicuro:

- 1) l'esigenza di P1 non può essere soddisfatta
- 2) l'esigenza di P2 non può essere soddisfatta
- 3) l'esigenza di P3 non può essere soddisfatta
- 4) l'esigenza di P4 non può essere soddisfatta

LO STATO NON E' SICURO

ALGORITMO DEL BANCHIERE CON RISORSE MULTIPLE DI PIU' TIPI (2.1)

Un sistema con processi P1, P2, P3, P4 e risorse dei tipi R1, R2, R3, R4, rispettivamente di molteplicità [4, 5, 5, 5], i processi dichiarano inizialmente le seguenti esigenze:

ESIGENZA INIZIALE				
	R1	R2	R3	R4
P1	2	3	1	1
P2	2	0	1	3
P3	0	1	0	2
P4	0	2	3	2

Stato raggiunto dal sistema al tempo t :

ASSEGNAZIONE ATTUALE →	MOLTEPLICITA' →					R1	R2	R3	R4	
		R1	R2	R3	R4	4	5	5	5	
	P1	2	1	1	1					
	P2	2	0	1	2					
	P3	0	1	0	0					
	P4	0	2	2	2					
DISPONIBILTA' ATTUALE		R1	R2	R3	R4					
		0	1	1	0					
						ESIGENZA ATTUALE				
						P1	0	2	0	0
						P2	0	0	0	1
						P3	0	0	0	2
						P4	0	0	1	0

Stato raggiunto dal sistema al tempo t :

Al tempo t processo P1 richiede una risorsa di tipo R2: stato raggiunto dopo l'ipotetica assegnazione:

P. Ancilotti, M. Boari, A. Ciampolini, G. Lipari - Sistemi Operativi

Al tempo t processo P1 richiede una risorsa di tipo R2: stato raggiunto dopo l'ipotetica assegnazione:

Verifica dello stato sicuro:

P. Ancilotti, M. Boari, A. Ciampolini, G. Lipari - Sistemi Operativi

ALGORITMO DEL BANCHIERE CON RISORSE MULTIPLE DI PIU' TIPI (2.4)

Verifica dello stato sicuro:

1) l'esigenza di P4 può essere soddisfatta e P4 può terminare

		MOLTEPLICITA' →								
		R1	R2	R3	R4	R1	R2	R3	R4	
ASSEGNAZIONE ATTUALE →	P1	2	2	1	1	4	5	5	5	
	P2	2	0	1	2					
	P3	0	1	0	0					
	P4	-	-	-	-					
DISPONIBILTA' ATTUALE		R1	R2	R3	R4					
		0	2	3	2					
						ESIGENZA ATTUALE				
						P1	0	1	0	0
						P2	0	0	0	1
						P3	0	0	0	2
						P4	-	-	-	-

Verifica dello stato sicuro:

1) l'esigenza di P4 può essere soddisfatta e P4 può terminare

2) l'esigenza di P1 può essere soddisfatta e P1 può terminare

		MOLTEPLICITA' →								
		R1	R2	R3	R4					
		4	5	5	5					
ASSEGNAZIONE ATTUALE →	P1	-	-	-	-					
	P2	2	0	1	2					
	P3	0	1	0	0					
	P4	-	-	-	-					
DISPONIBILTA' ATTUALE		R1	R2	R3	R4					
		2	4	4	3					
						ESIGENZA ATTUALE				
						P1	-	-	-	-
						P2	0	0	0	1
						P3	0	0	0	2
						P4	-	-	-	-

Verifica dello stato sicuro:

- ASSEGNAZIONE ATTUALE →

	R1	R2	R3	R4
P1	-	-	-	-
P2	2	0	1	2
P3	0	1	0	0
P4	-	-	-	-

DISPONIBILTA' ATTUALE

R1	R2	R3	R4
2	4	4	3

MOLTEPLICITA' →

R1	R2	R3	R4
4	5	5	5

ESIGENZA ATTUALE

	R1	R2	R3	R4
P1	-	-	-	-
P2	0	0	0	1
P3	0	0	0	2
P4	-	-	-	-

Verifica dello stato sicuro:

- ASSEGNAZIONE ATTUALE →

	R1	R2	R3	R4
P1	-	-	-	-
P2	-	-	-	-
P3	0	1	0	0
P4	-	-	-	-

DISPONIBILITA' ATTUALE

R1	R2	R3	R4
4	4	5	5

MOLTEPLICITA' →

R1	R2	R3	R4
4	5	5	5

ESIGENZA ATTUALE

	R1	R2	R3	R4
P1	-	-	-	-
P2	-	-	-	-
P3	0	0	0	2
P4	-	-	-	-

ALGORITMO DEL BANCHIERE CON RISORSE MULTIPLE DI PIU' TIPI (2.6)

Verifica dello stato sicuro:

- 1) l'esigenza di P4 può essere soddisfatta e P4 può terminare
- 2) l'esigenza di P1 può essere soddisfatta e P1 può terminare
- 3) l'esigenza di P2 può essere soddisfatta e P2 può terminare

		MOLTEPLICITA' →							
		R1	R2	R3	R4	R1	R2	R3	R4
ASSEGNAZIONE ATTUALE →	P1	-	-	-	-	4	5	5	5
	P2	-	-	-	-				
	P3	0	1	0	0	P1	-	-	-
	P4	-	-	-	-	P2	-	-	-
						P3	0	0	2
						P4	-	-	-
						ESIGENZA ATTUALE			

Verifica dello stato sicuro:

- 1) l'esigenza di P4 può essere soddisfatta e P4 può terminare
- 2) l'esigenza di P1 può essere soddisfatta e P1 può terminare
- 3) l'esigenza di P2 può essere soddisfatta e P2 può terminare
- 4) l'esigenza di P3 può essere soddisfatta e P3 può terminare

STATO SICURO !

		MOLTEPLICITA' →							
		R1	R2	R3	R4	R1	R2	R3	R4
ASSEGNAZIONE ATTUALE →	P1	-	-	-	-	4	5	5	5
	P2	-	-	-	-				
	P3	-	-	-	-	P1	-	-	-
	P4	-	-	-	-	P2	-	-	-
						P3	-	-	-
						P4	-	-	-
						ESIGENZA ATTUALE			

Stallo

Riconoscimento ed eliminazione

- **Riconoscimento dei processi in attesa circolare**
 - Tramite stato sicuro o tramite grafo di allocazione delle risorse
- **Soppressione di uno o più processi in attesa circolare**
 - recupero delle risorse
- **Rilascio forzato di alcune risorse**
 - Check-pointing e roll-back

(le risorse non sono prerilasciabili: necessario ripristinare uno stato consistente)

Eliminazione dello stallo

- **Soppressione di uno o più processi**

È il modo più semplice, sebbene drastico, per eliminare lo stallo

- Si forza la terminazione di uno dei processi in stallo
- Le risorse recuperate dal processo terminato possono essere assegnate agli altri processi sospesi

**Scelta del processo da sopprimere:
criterio di “minimo danno”**

Eliminazione dello stallo

Rilascio forzato di risorse

- forzare il rilascio di una o più risorse da parte di uno dei processi in stallo

Le risorse non sono prerilasciabili: il sistema può portarsi in uno stato inconsistente

Sono necessari:

- Check-pointing
 - Periodico salvataggio di stati consistenti
- Rollback
 - Si ripristina un precedente stato di check-point
 - Si fa ripartire il processo